ABSTRACT
The recent globalization of markets, technology and competition has increased business’ needs in terms of flexibility, quality, cost-effectiveness and timeliness. Information technology (IT) has played an important role in this field as a way of answering to these requirements. With the increased use of the Internet, electronic business is one area of IT that has been dramatically affected in the past decade, and so, electronic commerce has transformed and became one of the most important forces shaping business today.

Several requirements affect and condition the development of an electronic invoice solution. From B2B necessities, through legislation and contemplating a specific technological scenario, this dissertation aims to automate the invoicing process between two companies using BizTalk Server platform to provide the means to answer this business process’ requirements and afford further improvements in terms of supported formats and integration with other business support systems.

Keywords
Electronic Invoice, XML, UBL2.0, BizTalk, AS2, Advanced Electronic Signatures, BAM, SAP

1. INTRODUCTION
Many companies still have their invoicing processes semi-automated, typically the flow begins with the delivery of goods or services, and the creation of an invoice in the ERP system. After this first phase, it is necessary to print an invoice that will be sent to the buyer via post mail or fax.

Once the invoice is received, it passes through a phase of manual work where it gets digitalized before entering the buyer’s information system (IS), going eventually through the finance department where an order to pay the seller is emitted. The activities that start with the creation of the invoice at the seller company, until the manual integration at the buyer company, produce a major payload in terms of time spent in handling and processing of invoices, costs with paper usage and postage, etc.

With this in mind, companies are turning into process optimization and automation. This invoicing process as special features, allowing us to foresee great possibilities in the near future in terms of process optimization and automation. In part this is possible due to the manual workflow involved, approval processes, and some legal aspects that need to be fulfilled. And on the other hand, it is related with technological advances in ERPs and integration engines that ease the compliance with legal matters, and provide better support for automation of processes.

The scope of this dissertation is therefore the automate the invoicing process having into account existing systems and processes at both the seller and the buyer company.

This dissertation sets the following objectives:

• Achieving autonomy, decoupling and heterogeneity by using the SOA approach and by using standard XML message formats that can guarantee the interoperability between systems;

• In technological terms, the main challenge of this dissertation is to explore the functionalities of BizTalk Server 2006 R2 and SharePoint Services 3.0, and to find the limitations that these two main technologies have that can affect the developing of an invoicing process. One important aspect is that we want to minimize the effort of the developer by using as much of-the-shelf functionalities as possible and only using APIs when absolutely necessary. When limitations are found, solutions should be presented and implemented to verify their validity.

• In order to enrich this work, one important objective is to provide a mean to explore business data used on the daily basis. Using Business Activity Monitoring we want data that is stored in the invoice payload to be collected from the message engine and shown in a relevant manner in order to produce information that can be used in business intelligence.

• Finally one big objective is related with a specific technological scenario; where we want to integrate with a SAP R/3 system add some extra value besides the capacity of creating Electronic Invoices, like for instance the implementation of AS2 exchange protocol, implementation of Business Activity Monitoring allowing data exploration and the creation of summary maps, and the implementation of the means necessary to search and audit the stored and traded invoices.

2. STATE OF THE ART REFERENCE ARCHITECTURE FOR ELECTRONIC INVOICING SYSTEMS
This reference architecture is based in the analysis of generic frameworks and implemented architectures of electronic business solutions (SELIS and Netdocs) transposed into the domain of our problem, i.e. B2B electronic invoicing, always
building on top of well accepted past work and standards that could support future work.

Besides this, the reference architecture is supported by the Web Services Framework introduced by Huang and Chung [1], on the reference system for internet based inter-enterprise electronic commerce by Shina and Leem [2], and with a mix of our own personal thoughts and concepts gathered from the vast reference architectures proposed in Enterprise Integration - The Essential Guide to Integration Solutions [3].

The objective of this reference architecture is to provide the means to develop an independent software component that can be used as a complement to the existing architecture layout, integrating two or more applications in a B2B environment. Nevertheless it is important to have in mind that nowadays there are several service providers that guarantee compliance with these requirements. This reference architecture also provides an alternative approach which is point-to-point connection using a third-party intermediary.

Figure 1 presents the eFact reference architecture. The outstanding part of this section explains each layer and how each intra layer component interact from a holistic point of view.

Figure 1: eFact reference architecture.

2.1 Interface Layer

This layer is the main entry point for the architecture. The objective is to have different technology platforms connecting to this layer so it acts like an abstracting coat to all outside users, should they be a user connected to a web browser, a legacy application, or an ERP solution.

It provides the means to:

- Present data in a structured way for users or other applications in a pre-existing architecture to consume;
- Extract data from other existing applications.

2.2 Enterprise Service Bus

Next to the interface layer, an enterprise service bus (ESB) can be used, allowing a more flexible way to spread the functionality along the layers, allowing the establishment of a service infrastructure that supports the use of standards like XML and Web Services, enabling for instance content-based routing.

The ESB is a good solution when Web Services are used as the method of connectivity to other applications [3]. Since we also must contemplate this situation as a possible business integration option, we believe this is a good way to provide the infrastructure for the development and management of these services.

Given that on one hand we might want to integrate directly with an application through a WebService, and on the other hand, have a nice and user friendly interface to do a similar task based also on Web Services, the use of such technology enables an environment where change can come from multiple channels.

2.3 Process Orchestration Layer

Here resides the main functionality of the solution. On one hand, we must have core services that are responsible for the proper electronic invoice certification (electronic invoice content validation, electronic signature, etc.), and on the other hand we must have some integration business logic services that make use of the well defined core services, in order to address specific business scenarios.

Other main subject that we should address is the message format translation. In order to have a more flexible and adaptable architecture, we should have into account two concepts:

- The use of a well defined message format for internal computation.
- The use of a repository of translation models.

2.4 Storage Layer

At a lower level is the storage layer where we propose three types of repositories:

2.4.1 Electronic Invoice Archive

This is the main archive where information regarding electronic invoices should be stored. Here the PAIN properties should be addressed with special attention in order to guarantee a high level of security.

2.4.2 Data format translation models

With a well defined internal message format it is possible to develop adapters/translators for in/out message flow, allowing integration with other systems and their internal formats. This is also very important for audit purposes, where typically it is necessary to export the data in a specific format for the audit entities.

2.4.3 Configuration

This repository can be seen as container where all information about configurable components is stored. This includes trading policies, configuration about electronic signatures, encryption algorithms, etc.

2.5 Common Services Layer

Strategically vertically placed, the common system services contemplate Business Activity Monitoring (BAM), Audit and Security at all layers of the reference architecture, providing real-time reporting on the business transactions.

Also very important, here resides the security functionality, access control mechanisms, support for audit procedures, etc. The objective is to have these services available at each layer of the reference architecture and plugged into the enterprise service bus, so that changes into the security policies and audit mechanisms can take effect and become accessible at each layer.

\[1\] PAIN properties: Privacy, Authenticity, Integrity and Non-repudiation.
3. PROBLEM DEFINITION

Process automation is a recurring problem in today’s businesses; companies are seeking to provide solutions to these problems as a means to restructure aging bureaucratized processes in an attempt to achieve the strategic objectives of increased efficiency, reduced costs, improved quality, and greater customer satisfaction [4].

In this case we focus on the automation of the invoicing process between two companies (B2B) on the Portuguese market. Thus problem we have in hands resumes to a few aspects that need to be addressed, they can be grouped into three distinct domains:

- The challenges imposed by B2B interactions;
- Requirements imposed by legislation;
- Specific technological requirements.

3.1 The challenges imposed by B2B interactions

Electronic commerce is not simply about business transactions that run over the Internet, but is fundamentally about the flow of information [5]. Nowadays the boundaries of organizations are more relaxed than they used to be and this propels the automation of relationships with business partners.

As a consequence, alliances in which businesses join their applications, databases, and systems to share costs, skills and resources in offering value-added services, are being formed. The ultimate goal of B2B electronic commerce is therefore to have inter- and intra-enterprise applications evolve independently, yet allowing them to effectively and conveniently use each other’s functionality [6].

Interactions in B2B electronic commerce offer unique challenges because of issues such as scalability, autonomy, heterogeneity, coupling and dependability among partners and security. B2B electronic commerce requires the integration and interoperability of both applications and data. Disparate data representations between partners’ systems must be dealt with. Interaction is also required at a higher level for connecting (i) front-end with back-end systems, (ii) proprietary/legacy data sources, applications, processes, and workflows to the Web, and (iii) trading partners’ systems [6].

3.2 Requirements imposed by legislation

In the Portuguese case, there are a set of crucial aspects that every electronic invoice solution must address:

- Invoice Format;
- Invoice Integrity;
- Invoice Confidentiality and Privacy;
- Authenticity of origin;
- Non-repudiation of the origin and the destination;
- Invoice Archive;

3.3 Specific technological requirements

The final domain we must address is related with the specific technological requirements. More often than not, integration projects have to deal with existing IT architectures and existing software.

In this dissertation one of the requirements, in addition to the development of a SOA solution based on BizTalk Server, was the integration with a SAP environment using the BizTalk’s pre-packaged SAP Adapter. The main challenge resides in the understanding of BizTalk and SAP interoperability: which mechanisms are available, in which format the messages need to be exchanged, the necessary SAP and BizTalk configurations, etc.

4. SOLUTION PROPOSAL

The work to be done needs to provide an answer to two aspects: a solution to the invoicing problem from a theoretical point of view, and technology interoperability that provide the necessary support to these solutions.

4.1 Service Architecture

To better structure our proposal a Service Architecture was developed having into account a set of business events identified. These business events represent the activities that the solution must support.

For each business event identified, we created a description that explains in what it consists and how it is initialized. We also analyzed what responses were needed to answer to the business event in a correct manner.

Then the system responses defined in the Business Events Table were used to determine the essential services the system must provide. One important matter, which we can see in this dissertation, is that some of these services already exist as part of the technology that we shall use.

4.2 Invoice Format

Having into account that the format of business documents is crucial in terms of interoperability between business partners, we must adopt a specification that:

- Provides support for the invoicing information;
- Has flexibility in terms of evolution and introduction and/or removal of invoicing information;
- Is suited for exchange over the internet;
- Supports easy transformation between formats;
- Is independent of the context.

Being in the Portuguese market, and as proposed by UMIC2, we decided to use UBL 2.0 specification to address the invoicing problem. The existence of a unique internal format makes it easier to integrate with backend systems, and allows the development of an intermediate layer of services that do all the necessary transformations between received documents and sent documents.

The development of an intermediate transformation layer provides the means for a unique core processing format after
the arrival of an invoice (at the integration layer). These transformations will be done using Extensible Stylesheet Language Transformations (XSLT), which are supported by BizTalk Server.

So why is an internal format an advantage? Figure 2 exemplifies. If we had n input and output formats, we would have at least n(n-1) transformations [7] to perform (greens arrows only determine the flux of information through the core processing component and not an actual transformation). Adding to this transformation process we would have a highly complex core processing for the electronic invoices.

![Figure 2: N to N transformations without internal format.](image)

On the other hand, if we had an internal format, we would reduce the number of transformations to 2n [7], but more important than that, we totally simplify the core processing of the electronic invoice, and that is the main conclusion we can extract from this section.

This also provides the flexibility so that in the future more formats can be added with a minimum impact in the global solution. The bottom line here is that we’re aiming at a clean and simple solution.

### 4.3 Electronic Invoice Archive and Process Visibility

In terms of the proposed solution for archival we will use Windows SharePoint Services (WSS) as the main platform. WSS provides an Enterprise Service Bus with all the necessary services for storage and security, allowing a good alignment with the SOA objective.

The final goal is to have a storage layer with specific company repositories and where invoices will be stored in three categories: first, by company specific repositories, then by sent or received transactions with other partners, and finally by a separation of signed invoices and to visualize invoices. This separation allows different visualization methods using InfoPath templates, according to user desire.

Another important matter is that we don’t have to create a domain model from scratch to store information. WSS provides, with the required administrator privileges, an out of the box interface that allows a fully customizable repository, creation of access control policies using groups, users and actions over objects.

Also, WSS will provide the interface and visibility to all the invoicing process through BAM Portal and integration with BizTalk’s messaging engine. As soon as an digital invoice arrives (by web service invocation, file system, or direct integration using adapters), the core processing component will archive the now electronic invoice in the specific company repository and update, through the Business Activity Monitoring activities, the views over business data consumed and produced by the process, as well as business events, real-time business intelligence (BI), predefined reports of the summary map and AS2 non-repudiation data.

### 4.4 Invoice Integrity and Authenticity

The integrity and authenticity of the documents must be guaranteed during the transactions between business partners. For this UMIC recommends the use of S/MIME signatures with the algorithm RSA and with SHA-1 for hashing purposes.

S/MIME uses PKCS (Public Key Cryptographic Standards) to provide the mechanism for digital signatures and data encryption. In order to sign and/or encrypt a MIME message, at least one public/private key pair is needed. The public key is provided to users with whom secure communication is desired.

The sender’s private key is used to digitally sign a MIME message. When this message is received by the recipient, he uses the sender’s public key to verify the digital signature. For encryption, the sender uses the recipient’s public key to encrypt the MIME message. When the message is received by the recipient, he uses his own private key to decrypt the message.

To use S/MIME, digital certificates are needed. UMIC recommends the use of X509 certificates that must be obtained from a certificate authority; if not trading partners should agree and self-certify each other.

The UMIC recommendations fall in perfectly with BizTalk, in the way that they are totally supported by this technology. In fact BizTalk has pipeline components that allow a total automation of the signature process using S/MIME.

### 4.5 Non-Repudiation

An electronic invoicing solution must also guarantee that in a transaction the relations between the sender, the recipient and the invoice are maintained for audit and traceability reasons. Once again UMIC recommends the use of the Applicability Statement 2 (AS2) specification to transport data securely and reliably over the Internet.

The AS2 protocol is based on HTTP and S/MIME. Files are sent as "attachments" in a specially coded S/MIME message (an AS2 message). AS2 messages are always sent using the HTTP or HTTPS protocol, Secure Sockets Layer (SSL) is implied by HTTPS, and usually use the "POST" method. AS2 messages can be signed and/or encrypted, but do not have to be.

To transport data securely and reliably over the internet, AS2 uses the MDN (message disposition notification) mechanism that basically acts like a receipt that proves that the sending and receiving of the invoices really took place, and which parties were involved. The AS2-MDN exists in two varieties: synchronous and asynchronous [8].

The advantage of the synchronous MDN is that it can provide the sender of the AS2 Message with a verifiable confirmation of message delivery within a synchronous logic flow.

In terms of security X.509 certificates are required [8]. It is recommended that trading partners self-certify each other if an agreed-upon certification authority is not used. This applicability statement does not require the use of a certification authority. The use of a certification authority is therefore optional. Certificates may be self-signed. In
terms of implementation this was the path chosen in order to simplify the development of the solution.

Figure 3 exemplifies the process of sending and receiving AS2 messages in a scenario where the BizTalk server is used as integration engine.

Invoice information is extracted from an ERP system and is certified by the invoicing process orchestrated in BizTalk Server. At the end of this process the Electronic Invoice is enveloped in an AS2 message and sent to the trading partner. At the same time BizTalk updates its own non-repudiation database.

Upon the receipt of the message and its successful decryption or signature validation (as necessary) an MDN "success" message will be sent back to the original sender. This MDN is typically signed but never encrypted (unless temporarily encrypted in transit via HTTPS).

Upon the receipt and successful verification of the signature on the MDN, the original sender will "know" that the recipient got their message (this provides the non-repudiation element of AS2) and can thus terminate the transaction on the non-repudiation database.

4.6 Technological Interoperability

Figure 4 depicts a view of the platforms and software components that will be used. BizTalk Server will be used as an orchestration engine that will define the necessary activities for Electronic Invoice certification, and secure communication between trading partners.

The message broker will be supported by SQL Server and Windows SharePoint Services for archive purposes. Additional interoperability with the Office System is also supported through InfoPath and Excel. Eventually Portable Document Format (PDF) electronic invoices will also be generated by BizTalk Server for email delivery or notification.

BizTalk Server will receive invoices in a digital format and will apply all the necessary transformations, answering all necessary legal requirements before forwarding the (electronic) invoice to another system.

In terms of business support systems we will only focus on the integration with SAP R/3, since there was some interest, in terms of research and implementation, from the company where this dissertation was realized.

5. IMPLEMENTATION

A general overview of the Orchestration Architecture in the BizTalk solution is given in Figure 5.

The solution is separated in a send side and a receive side logics.

- **Send side**
  Set of orchestrations that are responsible for extracting digital (uncertified) invoices from other systems (through FileSystem, Web Service invocation or direct integration with a SAP system), and for the execution of a series of activities that certify the digital invoice in legal terms (transforming it into an Electronic Invoice) and sending it to the receive side of the solution.

- **Receive side**
  Set of orchestrations that are capable of receiving Electronic Invoices and are intended for integration with other systems. This receive side can be located on a remote machine.

The justification for this separation is the creation of a solution that can operate exclusively using the send side logics, extracting information from an existing system and certifying it into a local repository and operating with a third party service provider, or if we need interaction between two different systems, we can use the receive side logics as a receive adapter that can interoperate with the send side.

In an analogy with the Service Architecture, we can identify where each main service category is located:

- Integration services (integration logic orchestrations);
- Transformation services (core entry and exit points);
- Core services (core orchestrations);
- BAM services (between the two core logics).
5.1 Methodology

The development methodology was based on two main approaches: incremental development and the use of proof of concepts.

The incremental development is a scheduling and staging strategy allowing portions of the system to be developed at different times or rates, and integrated as they are completed [9].

A proof of concept is a short and/or incomplete realization of a certain method or idea to demonstrate its feasibility, or a demonstration whose purpose is to verify that some concept or theory is probably capable of exploitation in a useful manner. It is usually considered a milestone on the way to a fully functioning prototype.

In this way, the implementation methodology started with the definition of a series of milestones regarding the requirements that needed to be answered and the Service Categories defined:

- Receive a invoice in a generic format and convert it to an internal format (UBL 2.0);
- Receive a invoice, convert and sign it;
- Store the signed invoice in the archive in two formats (original signed format, and visualization format);
- After storage, send the electronic invoice via AS2 to the business partner;
- Receive an electronic invoice via AS2, validate its signature and store it in the partner archive;
- Apply business activity monitoring (BAM) to all the solution;
- Integrate with SAP R/3.

After each milestone was reached, all the solution was tested and verified for conceptual errors in terms of global invoicing process execution.

Then the output of each proof of concept was integrated with the existing solution. This allowed the development of a system piece by piece and permitted additions to the requirements as well as improvements in the development process.

5.2 Milestone 1 - Implementation of the UBL 2.0 Format

In order to complete this first milestone it was necessary to know what kind of information exists in a typical paper invoice. To reach this end it was necessary to ask UMIC for the legal information requirements and which UBL 2.0 fields are necessary to hold the invoicing information.

UMIC supplied a set of information that contain the necessary fields and a brief description on the invoicing information, as well as the validation rules in order to verify the syntax and semantics of used data. The most essential element of this information is that it provides the mappings to UBL fields.

As shown in Figure 5 the orchestration architecture is composed by six main orchestrations, this Milestone focused on the development the "Core Entry Point". This orchestration receives the digital invoice in any XML format and will test the format’s namespace. The namespace will be the key element that will differentiate the various supported XML formats.

Figure 6 exemplifies the decision process that starts with the reception of a XML formatted message.

![Figure 6: Core Entry Point Decision Point.](image)

After detecting the XML format a XLST transformation is applied. This transformation is done using BizTalk’s XML mapper that provides a user interface and generates the XSLT code behind. The XSLT transformation copies the values from a source XML schema into a destination XML schema (UBL 2.0).

In terms of extensibility to other formats, this solution simply requires a new decision branch and the creation of a new XLST. All this is possible due to the advantages of the use of a well defined internal format.

5.3 Milestone 2 - Implementation of Electronic Signatures

UMIC recommended the S/MIME signature type for the electronic invoice solution. First of all, it was necessary to get certificates for test purposes. Since it wasn’t easy to contact a certification authority to help us in this issue, we used the Windows Certificate tool to generate a self signed X509 certificate from which we extracted the public and private keys to use in our solution.

The private keys were stored in the Windows Personal Store and the public key in the Windows Other People Store. This is where BizTalk searches for certificates when defining the administration properties.

BizTalk provides built-in mechanisms to encode and decode messages using S/MIME. Within BizTalk, the signature process is usually done in the pipeline processing using specific pipeline components. One must create a BizTalk project where a message is sent using a send port with a custom send pipeline. This pipeline must contain an S/MIME encoder that has as a property a certificate thumbprint which is then used to sign the XML message. The downside of this implementation is that the thumbprint must be hardcoded in development time.

Since we had the requirement of signing the electronic invoices using different certificates from each company, this situation posed as a serious drawback because we didn’t wanted to design new pipelines or change these properties each time a new company is added to the solution or each time an Electronic Invoice is sent from a different company.

Due to this BizTalk limitation, we had the need to find a way to assign to each company a certificate in order to univocally identify the owner of each electronic invoice. After consulting Microsoft Portugal on this issue, we came up
with a solution that consists in a custom pipeline component that makes use of the AS2 Party properties, allowing to configure each company’s certificate at runtime.

Figure 7 illustrates the logic behind the custom pipeline component.

As the XML message passes through the pipeline component, routing information based on the content of the invoice is extracted and used to determine which party property must be accessed to get the signature certificate. Once the signature certificate is obtained, its thumbprint is extracted and used in the signature process.

5.4 Milestone 3 - Implementation of the Archive

The archive component of the solution was implemented in integration with Windows SharePoint Services 3.0 (WSS). This integration was realized through the WSS service bus. The service bus provides the flexibility necessary to guarantee a good decoupling between these two main technologies.

In terms of implementation, the interactions with the archive are part of the core functionality and so they reside in the core orchestrations.

After the execution of the custom send pipeline (that signs the invoice), the Electronic Invoice is archived in the local repository. For this, routing information is extracted from the invoice, namely the sender company name and the invoice identification. Then this information is used to create the necessary storage structure (as described in the solution proposal chapter).

The Electronic Invoices are stored in two formats:

- Unsigned InfoPath XML format (to visualization)
- Signed XML format

The InfoPath XML is created inside the core orchestration by adding a processing instruction (PI) that tells SharePoint that this file should be used with an InfoPath application. InfoPath allows the creation of custom made forms that are then used as templates in SharePoint Libraries. The templates take as source data the information contained in the XML Electronic Invoices, and then apply an InfoPath form that allows better legibility and understanding of relevant information. With these two steps (adding a pre-instruction and creating a visual template) it is possible to develop various Electronic Invoice views based on specific user preferences.

The other storage format is S/MIME signed XML format. For interoperability reasons, BizTalk appends the signature to the original message and then encodes automatically all signed messages in Base64.

These messages are created from the same information source as InfoPath XML, and so they are uploaded simultaneously to the repository after being sent to the trading partner.

Nevertheless the InfoPath Electronic Invoice does not have any kind of legal value and is intended only for visualization purposes. As so, the only legal and valuable information is that stored in the signed invoices.

The storage is done through the consumption of Web Services in the WSS service bus, all interactions between WSS and BTS are done using this service bus, see Figure 8.

In terms of security, WSS provides its own access control mechanism based on groups, users and actions over objects. Since WSS also provides the interface for the solution, we can have an integrated access control and guarantee the security of the stored information.

5.5 Milestone 4 - Implementation of AS2 Communications

This milestone consisted in the implementation of the AS2 protocol to answer the non-repudiation requirements. This implementation was realized using the new BizTalk Server capabilities in this area. BizTalk provides the adapter and a non-repudiation database where all AS2 transactions are logged.

AS2 communications are also part of the core functionalities of the general solution, and so, they are implemented as part of the core orchestrations, more specifically as the last activity in the Core Logic on the send side and the trigger to the Core Logic in the receive side.

To setup an AS2 connection between two partners using BizTalk it is necessary to configure the AS2 party properties in BizTalk Administration Console. Here we need to specify a party alias to be used for routing as filter expression on send ports and a certificate that is used to uniquely identify each party (trading partner). This certificate is the one used in the signing of the Electronic Invoices. These properties also define whether or not this AS2 implementation uses the MDN mechanism.

Figure 9 illustrates the general interaction between two parties.

After all the core processing, the message is sent to the BizTalk’s message box where a send port picks it up (based on AS2 alias filter expressions) and passes it through an AS2 send pipeline. The pipeline envelopes the original Electronic Invoice in a S/MIME message, adds to the message header
the source and destination for tracking and routing purposes (AS2-From and AS2-To properties) and sends the final message on AS2 over HTTP to the HTTP receive location.

In this process, BTS logs in the non-repudiation database the start of the transaction. When the message arrives at the destination the HTTP receive port receives the message and passes it through the AS2 receive pipeline, the MDN mechanism sends the proof of receipt and the transaction is concluded in the non-repudiation database (MDNs are also stored in the non-repudiation database).

5.6 Milestone 5 - Implementation of the Receive Side

The receive side is like a symmetric copy of the send side orchestrations. In fact, it was developed almost at the same time as all the send functionalities.

Having this into account, these orchestrations can be used as an adapter that is always listening to the AS2 communications.

5.7 Milestone 6 - Implementation of BAM

Business Activity Monitoring (BAM) is a collection of tools that allow you to manage aggregations, alerts, and profiles to monitor relevant business metrics (called Key Performance Indicators or KPIs). It gives you end-to-end visibility into business processes, providing accurate information about the status and results of various operations, processes and transactions so that specific problem areas can be addressed and issues resolved.

BAM allows tracking specific business data along the invoicing process. Within BizTalk BAM is composed of activities and views over activities. There are two methods to apply BAM in BizTalk:

- Using Tracking Profiles
- Using the BAM API

Both were implemented and tested. The second one emerged as the most complete and flexible. Although required more manual work, the BAM API method is the one put in practice in the final solution.

Figure 10 depicts the two main BAM development processes:

As we can see there are a few steps in common. First we must define activities and views through Bam.xls, an Excel Stylesheet that is used to define the business data we want to monitor. Bam.xls is typically located on <BizTalk Server Installation Path>\Tracking\Bam.xls and must be copy to the solution’s project folder so that it can be changed as needed.

Instances of BAM activities contain the data that is used to generate the Key Performance Indicator (KPI) reports. When you define a BAM activity you specify the milestones and other data to collect from the application that is being monitored. These data items are called activity items. After you define the BAM activity and the activity items, you can use the activity to define KPI reports (called "views").

When defining these views, we are also defining cubes that allow excellent data exploration. Views make use of dimensions and measures based on activity items and can also span multiple activities.

A BAM view identifies the BAM activities and activity items that will be used to create KPI reports. A dimension identifies how data will be grouped in reports. You can think of a dimension as a row or column heading in a report. A measure identifies the data that will be computed and displayed in the reports. You can think of a measure as the cells in a report.

The problem with the tracking profile method is that it doesn’t support mapping data with repeating fields. This problem occurred when testing with invoices with multiple lines of products/services.

Since this functionality is crucial, a solution was required. After some research we developed a custom pipeline component that makes use of the BAM API. This BAM pipeline component allows to programmatically control the data we want to monitor. With this method we could reuse the activities and the views already defined and instead of mapping fields using tracking profiles the data was gathered when it passed through a send or receive pipeline. Figure 11 illustrates the data collecting process:

This component looks into the electronic invoice content
and starts a BAM activity, then it collects data, updates the BAM Primary Import database with fresh information and finally closes the activity when no more data is needed. This component allowed to iterate over each invoice line and extract the information we needed.

Finally now that all data could be collected, we needed some way to look into the information in a pleasant way. To extract BAM information, it is best to use a BAM view rather than going directly to the SQL tables in the Primary Import database. The view aggregates information from several tables to surface a complete view of the data. As so, BAM data can be explored using BAM Portal and WSS, allowing custom made queries to BAM database tables, drill-up and drill-down of data through the defined cubes in real-time using the real-time aggregations.

Finally we used BAM to address the summary map requirement. The elements obliged by legislation were modeled in BAM activities and two summary map views were created. Then relevant business data was monitored when it passed through a receive or send pipeline and finally explored using BAM Portal.

The conclusion of the BAM implementation is that although Microsoft says the best way to persist incoming data from a BizTalk process into BAM is to use the Tracking Profile Editor (TPE), we found that the BAM API offers more control and flexibility to the developer.

5.8 Milestone 7 - Integration with a SAP R/3 System

The first step to accomplish the integration with SAP was to establish a connection between the two systems before actually receive any kind of information from SAP.

The idea was to configure the two systems so that when an invoice is created in SAP, it automatically triggers a BizTalk orchestration. The orchestration would then undertake all necessary activities for the electronic invoice certification.

Since our aim was only to guarantee the integration between SAP and BizTalk, we shall not concern with the buying process that must be preconfigured in SAP in order to correctly generate invoices. After configuring SAP for IDoc outbound processing, we used a SAP tool that allows to generate IDocs, by specifying their message and base types, it also provides an interface to fill in IDoc fields, and then it allows to send IDocs to the recipient by specifying the partner no. (BIZTALK) and the partner type (LS - Logical System).

To prove the flexibility of the BizTalk SOA and the benefits of the SAP adapter, as well as the importance of the architecture design in terms of adoption of a new format (in the core entry point), the first step to conclude the integration was to use the SAP adapter to generate the SAP IDoc invoice equivalent XML schema (XSD), and to create a receive pipeline that uses a flat-file disassembler to convert from the IDoc flat-file to the IDoc XSD.

Once this first step was accomplished, we used the core entry point design to add a new decision branch for the new XSD format, as we saw in Figure 6.

Finally we added a receive location, that uses the pipeline created earlier, to the receive port of the send core functionalities.

Now when the SAP test tool for IDoc processing is used for outbound purposes, the IDoc is sent to BTS, on its arrival it is converted to the UBL format before it reaches the core orchestration, and then all activities are undertaken oblivious to the original invoice format.

6. RESULTS

6.1 Estimated Integration Effort

Solution cost in an integration project is usually measured by counting the number of integrations needed and the estimated cost of each integration. Besides this, another important characteristic is the licensing cost of the technology and whether or not there are adapters to the technologies involved in the integrations.

Assuming a simple scenario with two companies (Company A and Company B) exchanging invoices we would have at least two integrations, one of each send side functionality at each company. The cost of these two integrations would depend mainly on the technology involved. For instance, if both companies were using backend systems supported by the Line of Business (LoB) Adapter Pack shipped with BizTalk Server 2006, we would have an almost inexpensive integration effort in terms of implementation and deployment of the solution. Otherwise we would need the help of an in-house IT expert, with a minimum knowledge of the technology allowing medium effort integration.

On the other hand, we also need to have into account the integrations in the receive functionalities. If the receive side of the solution proposed in this dissertation is used, the integration effort would be out-of-the-box in terms of receiving and storage of Electronic Invoices. Eventually the only integrations needed would be with a company’s backend system if needed. Otherwise (if the receive side is not used) the effort would be more complex in the way it would be necessary to configure the AS2 properties accordingly with the other company’s AS2 adapter.

6.2 Performance

The performance in this work was a secondary objective. In fact, it wasn’t even modeled as a requirement. In this way, performance testing only refers to the observable timings and errors occurred in the processing of single invoices and in batch processing of multiple invoices.

After testing the solution using all developed interfaces we could conclude that the results of the FileSystem and the orchestrations exposed as Web Services are very similar in terms of processing time for each Electronic Invoice. Although the transmission time is not taken into account, the observed 7-10 second interval per invoice seems to be a very good result. The SAP adapter proved to be the fast and more reliable interface tested.

In terms of errors they were related with deadlocks in the BizTalk message engine. These kinds of faults are totally unpredictable from the developer’s standpoint and only started with a batch processing of 30 or more invoices.

Having into account that the HTTP transport times are missing due to local machine testing, we ought to expect that these numbers should be a little higher and more propitious to errors. Nevertheless we could conclude that the 20 Electronic Invoice mark for group processing is a good start point for the configuration of a batch execution environment.

6.3 Bottlenecks
Besides a simple performance analysis, we can also analyze the solution’s bottlenecks to gain better understanding of the activities that consume more time and that should be revisited in future work to improve overall performance.

In the send logics we observed that the more time spending activities are the core entry point and the core processing where we can see that most of the time goes to the application of the signature. The time spent on storage is minimal as it is done through Web Service invocation to SharePoint Services, putting all overhead away from the messaging engine. Finally we can see that other massive time consuming activity is the send and the receive AS2 functionalities that consume around 30% of the total processing time.

Regarding the receive logics we can observe that the core activities consume most of the time as expected, similarly the storage time is minimal and comparable with the times observed in the send side. One interesting aspect of the core receive side is that it consumes around 3.5 times less time that the core send time. This becomes obvious if we think that the send side has to apply a set of algorithms to sign the Electronic Invoice, and the receive side only has to validate the signature.

6.4 Ease of Integration, Expansion and Interoperability

Known as Service Oriented Architecture (SOA), the methodology is based on XML and Web services technologies and has been incorporated into Business Process Management and Enterprise Application Integration (BPM/EAI) platforms. The BizTalk Service Oriented Architecture primes in this field for its capacity of integration with other systems.

The SOA paradigm has redefined the concept of an application. An application is no longer an opaque, procedural implementation mechanism. Instead, it is an orchestrated sequence of messaging, routing, processing, and transformation events capable of processing the exposed declarative properties of rich (XML) documents. BPM/EAI platforms that incorporate the SOA paradigm are highly compelling because they provide numerous development and operational benefits which include the flexible loose coupling of components on a highly distributed basis, the addition, removal, and reconfiguration of any process activity or component without disrupting the process and the extensibility and reuse of both application components and entire applications.

In this work, these characteristics were explored in integration with a SAP R/3 system but easily systems from other vendors could be incorporated without affecting the general architecture. In fact, the BTS SOA allows the incremental development and the constant adding of new functionalities as services to this solution and from a holistic point of view, the enrichment of the existing service infrastructure as a whole. With well defined core functionality, this solution allows further development to focus on the extensibility in terms of integration with other back-end systems, new message formats and other types of electronic signatures without affecting the global processing.

7. CONCLUSIONS

The main objective of this dissertation was to develop, with the chosen technology, an Electronic Invoice solution that answers all legal requirements (according to Portuguese Legislation), and at the same time, follows the premises of the BizTalk Service Oriented Architecture. This objective was clearly accomplished and the final result is a solution with still very work to be done in order to improve its capabilities, but yet, already with a set of functionalities that are enough for the deployment of an automated invoicing process between two business partners.

To help us design our Orchestration Architecture it was defined a Service Architecture based on a set of identified business events. This Orchestrations Architecture allowed us to develop the services with a minimum code writing effort at the same time as functionalities could have their endpoints exposed as Web Services guaranteeing maximum interoperability with other systems.

The use of BizTalk technologies in this dissertation proved to be an excellent benefit as this message engine allowed to develop a series of processes in the form of orchestrations that coordinate a set of activities, implemented in a loosely manner just as SOA defends. This decoupling method defined a interface layer and allows that other applications or services that use these (BizTalk defined) services to worry only with WHAT the service does and not HOW the service does it.

8. REFERENCES


